Artifact Description: Studying the Impact of Power Capping on MapReduce-based, Data-intensive Mini-applications on Intel KNL and KNM Architectures

Student: Joshua Davis, University of Delaware
Advisors: Michela Taufer, Sunita Chandrasekaran, and Tao Gao

ABSTRACT

This artifact description present information to replicate the work in the ACM Student Cluster Competition poster titled "Studying the Impact of Power Capping on MapReduce-based, Data-intensive Mini-applications on Intel KNL and KNM Architectures" and authored by the undergraduate student Joshua Davis.

DESCRIPTION

1) Check-list (artifact meta-information):

- **Algorithm:** MapReduce wordcount
- **Program:** C++ library with MPICH
- **Compilation:** mpicc
- **Hardware:** Intel Xeon Phi 7250 (KNL) and Intel Xeon Phi 7295 (KNM)
- **Output:** KV and KMV counts, frequency, energy over time and frequency data
- **Experiment Workflow:** Install PAPI, install MPICH, install Mimir, clone Mimir-scripts, calibrate power limit, run tests, observe output
- **Experiment Customization:** Wordcount mini-app type (Map+Shuffle, GroupByKey, or ReduceByKey), hardware system (KNM or KNL), dataset unique words, communication buffer size
- **Publicly Available?:** no, will be made public if accepted

2) How delivered: If accepted, Mimir will be made available to clone from the following: https://github.com/TauferLab/Mimir-dev.git, and Mimir-scripts will be made available to clone at https://github.com/TauferLab/Mimir-scripts.git. PAPI can be cloned from https://bitbucket.org/icl/papi.git.

3) Hardware dependencies: This document covers only Intel KNL and KNM systems, but expanding to other platforms should not pose a significant challenge.

4) Software dependencies: Mimir needs MPICH 3.2 or above and a C++ compiler such as gcc. MPICH can be installed by these steps:

```
wget
http://www.mpich.org/static/downloads/3.2/ /
mpich-3.2.tar.gz
tar xvf mpich-3.2.tar.gz
cd mpich-3.2
./configure --prefix=$MPICH_INSTALL_DIR
make && make install
```

INSTALLATION

1) Clone PAPI. Then:

```
cd papi/src
./configure --prefix=$PAPI_INSTALL_DIR /
--with-components="powercap"
make && make install
make test
```

2) Add the MPICH bin directory at $MPICH_INSTALL_DIR to your $PATH, and add the PAPI lib directory at $PAPI_INSTALL_DIR to your $LD_LIBRARY_PATH.

3) Clone Mimir-dev and switch to the power-capping branch. Then, after moving to the new directory:

```
autoreconf -i
./configure --prefix=$MIMIR_INSTALL_DIR / --
with-papi=$PAPI_INSTALL_DIR
make clean
make && make install
```

EXPERIMENTAL WORKFLOW

1) Move to `$PAPI_DIR/src/components/powercap/tests/` and modify the powercap_limit.c test such that the last powercap setting (the line followed by the comment '// set power limit to plus 10 Watts') sets the power limit to the calibration limit. In our tests, this is 215 Watts, but ensure this is compatible with your system's Thermal Design power before setting. Run make clean and make to compile, and run the new powercap_limit to calibrate the powercap for your tests.

2) Clone Mimir-scripts and move to the linux-powercap directory. Configure a results directory appropriate to the code and run tests with:

```
./freq_test.sh dataset powercap /
buffersize_num duration(s) benchmark 1
```

where dataset is a recognized dataset setting in the test_minitest scripts (i.e., 4B50), powercap is a number from 1.0 to 0, buffersize num is a number of MB for communication buffer size from 8 to 200, duration is the estimated runtime in seconds, and benchmark is one of shuffle (Map+Shuffle), groupbykey, and reducebykey.

EVALUATION AND EXPECTED RESULTS

When running the Map+Shuffle and ReduceByKey mini-apps, the code outputs the following for each process:

```
[x]y: MapReduce: Map done. (KVs = z)
```

where x is the rank of the process, y is the total number of processes, and z is the number of KVs the Map function ends with. For Map+Shuffle, the sum of all z's (for every x) should be equal to the total number of words in the dataset. For ReduceByKey, this sun should equal the number of unique words.

When running the GroupByKey mini-apps, the following will output for each process:

```
[x]y: MapReduce: Reduce done. (KMVs = z)
```

The format is the same as above, except z is now the number of KMVs the Reduce process finishes with. For validation, the sum of all z's for every x should equal the number of unique words.

EXPERIMENT CUSTOMIZATION

For each test, all of the parameters described in Section 2 of the Experiment Workflow can be modified. In addition, we run each test for this poster on both KNL and KNM systems.